6.2 PLUS

The TRSDOS 6.2 Enhancement Package

DOSPLUS T.M. IV

Disk Operating System

for the TRS-80™ Model 4 Microcomputer

Micro-Systems Software Inc. announces **DOSPLUS IV,** the new alternative operating system for the TRS-80 Model 4. **DOSPLUS IV** ushers in a new age of efficiency and power for the Model 4, offering you the ultimate in features and convenience.

DOSPLUS IV is the culmination of all of our previous efforts in TRS-80 operating systems. It has all of the simple, user friendly operation of DOSPLUS 3.4 and all of the powerful, flexible capacity of DOSPLUS 3.5. And all of this with a touch of class.

DOSPLUS IV is truly programming elegance. The system has been written to be 100% media and program compatible with TRSDOS™ 6.0, so there won't be any problems transferring to the new system. **DOSPLUS IV** even uses the same version of MicroSoft BASIC (with certain enhancements, of course) to insure program compatibility.

From the DOS side, **DOSPLUS IV** features things such as automatic installation of ALL drivers (no need to use SET and FILTER to create "phantom" devices), global wildmask capacity on many library commands (ATTRIB, COPY, or REMOVE up to an entire disk at one time), piping and filtering (similar to systems such as UNIX and MS/DOS), and all of the same commands that you would expect from **DOSPLUS**.

On the BASIC side of things, there is an array sort utility to replace the CMD"O" that TRSDOS 6.0 takes away, a keyword space insertion function to save you hours of needless typing and editing, shorthand, INPUT@ (controlled screen formatting), and label addressing. **DOSPLUS IV** also causes BASIC to load and save programs many times faster than TRSDOS 6.0.

For the novice programmer that just wants the simplest, fastest, most reliable operating system they can have AND for the experienced programmer who wants the slickest, most powerful, device independent system they can have, **DOSPLUS IV** has it all! If this system sounds too good to be true, don't worry. It's better than that!

The **DOSPLUS IV** manual comes complete with a system technical reference manual (no extra cost options here). **DOSPLUS IV** includes disk and editing utilities and directory verification/repair programs that others want to charge you extra for.

Your operating system is a serious investment, treat it as a serious matter and get the only serious operating system for the Model 4; **DOSPLUS IV!**

Available from computer dealers everywhere or directly from:

Micro-Systems Software Inc.

4301-18 Oak Circle

Boca Raton, FL 33431

Sales: (800) 327-8724

Technical: (305) 983-3390

MicroNet: 70271,120 (SYSOPs of the Telcomm SIG)

Source: ST8719

Telex: 467384 MSS INC CI

6.0 PLUS™ - The TRSDOS 6.0 enhancement series

User's guide [Rev. A.1]

Table of contents

Section	Page
Overview	1
DOS Enhancements	1
DISKZAP	1
DISKDUMP	2
DIRCHECK	2
MAP	3
RESTORE	3
Summary	3
BASIC Enhancements	4
BE1 and BE2	4
Shorthand	4
Label addressing	5
Faster loads/saves	5
DI, DU, and DR	5
Extended error posting	5
OPTION statement	5
INPUT@	6
SORT	6
SR	7
REF	7
RESOLVE	8
Summary	8
Overall conclusion	8
DOS Utilities	9
Installation	9
DISKZAP	10
Set	10
Fill	12
Copy	13
Print	15
Verify	16
Format	17
Display	18
DISKDUMP	22
DIRCHECK	25
MAP	27
RESTORE	29
BASIC Enhancements	30
Installation	30
DI	31
DU	32
DR	32
SHORTHAND	33
REF	34
SR	35
SORT	36
INPUT@	39
Label addressing	41
Extended error posting	42
OPTION	43
RESOLVE	45

6.0 PLUS - The TRSDOS 6.0 Enhancement package

Welcome to the 6.0 PLUS overview! The purpose of this section of the documentation is to get you acquainted with the package you have purchased and give you an overview of the system combined with some practical ideas for usage.

This is NOT a step by step "how to". The amount of space available precludes treating the subject with that kind of detail. If you are seeking a reference work regarding disk repair, then we suggest that you refer to one of the more definitive works on the subject (and there are several for the TRS-80). The Model 4 TRSDOS 6.0 technical manual (that you may get from Radio Shack for an extra fee) contains an adequate explanation of diskette structure, directory organization, and file structure.

Let's take each of the utilities in turn and cover some tips and general operating concepts. We will also discuss some of the BASIC enhancements later.

This package is a series of enhancements to Model 4 TRSDOS version 6 and BASIC. These enhancements cover a wide range of areas from disk editing to BASIC programming.

The DOS enhancements

This set of enhancements include:

DISKZAP	Track and sector oriented disk editor
DISKDUMP	File oriented disk editor
DIRCHECK	Directory verification/repair utility
MAP	Disk mapper to locate files by track and sector
RESTORE	Re-activate files that have been REMOVEed

DISKZAP

The DISKZAP disk editor program provides for a wide range of functions. It is configurable to work on both floppy and hard disks and supports all varieties of possible formats.

DISKZAP's "verify" option allows you to completely verify all sectors of a disk for readability and will report all errors found. With the "format" option, you can format one track or an entire disk as needed. This can sometimes be the difference between losing a track's worth of information and losing the entire disk.

Perhaps the most often used option of DISKZAP is "display". This option allows you to display any sector on the disk and alter its contents directly. Many times this will allow you to install patches, repair blown sectors, or in other ways manipulate the disk when nothing else would work.

DISKZAP also features "copy", "fill", and "print" options. With copy, you can copy a series of sectors directly from one disk to another. Fill will allow you to write a specified data pattern into one sector or as many sectors as you want. Print gives you the ability to obtain hardcopy of any desired disk sectors.

In short, DISKZAP gives you the ability to directly work with the disks in a manner previously impossible. It can detect errors and aid you in repairing them. At the very least, DISKZAP's ability to repair a disk could mean the difference between losing two sectors of data or an entire disk.

DISKDUMP

Perhaps the only drawback to DISKZAP is that if you do not know EXACTLY where a file resides on a disk, you cannot edit that file. Many times all that is desired is to use the editor option on a particular file. This is where DISKDUMP enters the picture.

DISKDUMP has the same functions as the display option of DISKZAP. However, you don't have to inform it which track and sector to examine. Instead, you give it a file specification. DISKDUMP will locate the file in the directory and proceed directly to where it is stored on the disk.

Once it has displayed the first sector of the file, you may begin editing much the same way as you would with DISKZAP. You can even use DISKDUMP to search out a particular address in a load module format (machine language program) file. It is also possible to find any desired byte in a sector at the touch of a button.

One further advantage of DISKDUMP is that it may be used from within BASIC (if the 6.0 PLUS series BASIC enhancements have been installed). This enables the programmer developing software to pause, examine the actual disk sectors of their data files to make certain the information was written correctly, and return to BASIC with program and variables unharmed.

DIRCHECK

With DISKZAP, you can examine the sectors of a directory and ascertain that they are all readable, but the data there will be meaningless to the average user. You can be sure that you can read it, but how can you be sure that what you read is correct? With DIRCHECK.

DIRCHECK will examine the directory of any drive, hard or floppy, and report any error that it might find. It will check all areas of the directory, GAT (Granule Allocation Table), HIT (Hash Index Table), and file area (where the actual filenames and other data is stored).

If DIRCHECK reports no errors, you may be certain that your directory is intact. However, sometimes DIRCHECK will find problems. As you know, the directory is the most important track on the disk. Lose that, and you've lost all of the data in all of the files. Therefore, when DIRCHECK says something is wrong with the directory, what can you do?

Use the "fix" options of DIRCHECK. DIRCHECK will, on command, repair the directory to the best of its ability. So if you cannot figure out what all that confusing data in the directory sectors is all about, don't worry. DIRCHECK knows.

MAP

This utility will report the location of every file on the disk by track and sector. You can use this information in a variety of ways. If you have verified that a portion of a disk is bad with DISKZAP and you wish to see what files are affected, you can use MAP and scan the list of files looking for the one that was in the wrong place at the wrong time.

Also, if for some reason you desire to use DISKZAP on a file instead of DISKDUMP, you will need MAP to locate it for you. Since DISKZAP works only in hex and most people think in decimal, MAP will output in either. You may specify hexadecimal output for usage with DISKZAP or allow the standard decimal default if it makes it easier for you to visualize where the file is.

MAP will also list all segments of a file separately. Using this, you can determine: 1) if a file is in one piece or several pieces and 2) if those pieces are close together or spread all over the disk. If you determine that a file is in many segments scattered all over the disk and these segments are spaced far apart, then you may use the COPY command to copy the file to an empty disk and consolidate the data for better access times and less drive head travel.

RESTORE

Have you ever REMOVEed the wrong file? As soon as you do it, you get this sinking feeling as you hope that you had another copy somewhere. But there is no need to be concerned if you have the RESTORE utility handy.

When a file is REMOVEed, it is not actually erased from the directory. It is simply flagged as inactive, erased from the HIT, and the space it held is freed up in the GAT. However, if no other file has used its directory slot (over-writing the old entry) or used its disk space (over-writing the old data), the file can be rebuilt.

That's what RESTORE does. You supply the filename of the REMOVEed file and RESTORE will examine the directory to see if the entry is still there. If it is, and the disk space has not been re-allocated, then RESTORE will re-activate the entry, place it back into the HIT and once again allocate space to it in the GAT.

RESTORE is error proofed: If for any reason it feels that a file cannot be successfully brought back, it will inform you of this and abort. It will NEVER bring back one file at the expense of an active one.

Summary

This set of DOS enhancements provide functions that are sorely lacking in TRSDOS 6 for both the programmer and novice alike. They open up many areas of recourse if a disk "goes down". They help to make TRSDOS more powerful and user friendly. They help the Model 4 DOS make the Model 4 more of what it already is. The best buy in small computers.

The BASIC enhancements

MicroSoft BASIC is the finest around. An acknowledged industry standard. This set of enhancements will provide some functions that the novice will find a great shortcut and some more that the programmer will find they cannot work without. This set includes the following:

SORT	BASIC array sort facility
SR	Global BASIC text editor
REF	Comprehensive BASIC cross referencer
RESOLVE	Removes label addressing and inserts line numbers
BE1	General enhancement package #1
BE2	General enhancement package #2

BE1 and BE2

We'll cover these first since they are not so clear in the summary. These two programs contain a set of general enhancements that are actually merged with BASIC itself and become internal to the program as opposed to an external utility used while in BASIC.

Installation is simple. You just clear memory and load BASIC. Then you also load the selected enhancement package and dump BASIC back to disk using the addresses specified in the user's guide and you're done.

Why two packages? BE1 contains the INPUT@ controlled screen input option and BE2 does not. This allows you to decide whether you wish to use the extra memory for this option.

These enhancements provide a great range of additional functions, which we shall cover next, but they also provide the ability to use the DISKDUMP program (see DOS Enhancements) from BASIC and return to BASIC. The specific functions are:

Shorthand. This takes two forms; immediate commands and abbreviated statements. A brief summary:

Immediate commands

Key	Function
up arrow	List preceding line of program
down arrow	List next line of program
shift up arrow	List top line of program
shift down arrow	List bottom line of program
; (semi colon)	List top line of program
/ (slash mark)	List bottom line of program
. (period)	List current line of program
, (comma)	Edit current line of program

Abbreviated statements

Abbreviation	Statement
A	AUTO
D	DELETE
E	EDIT
G	GOTO
I	INPUT
K"	KILL"
L or L"	LIST or LOAD"
N	NAME
R or R"	RUN or RUN"
S"	SAVE"
!	SYSTEM

Also, when using this shorthand, typing spaces between statements (e.g. G 10) is not needed. "G10" will expand to "GOTO 10". Spaces and all. Perhaps the most useful abbreviation is "!" for "SYSTEM". It is much easier to exit BASIC by typing an exclamation mark and pressing ENTER. You may also use this when executing DOS commands (i.e. !"DIR").

Label addressing. After you have installed the enhancements to BASIC, you may implement label addressing in your programs. You define labels with the NAME statement (i.e. 10 NAME BEGIN). Later, simply reference that label as you would with any other line number (i.e. 100 GOTO TEST ... 2000 GOSUB TEST). Labels may be removed with the RESOLVE utility (covered later) for compatibility with standard BASICs.

You may LIST by label, EDIT by label, or use it any time that you might otherwise use a line number reference.

Faster load and saves. BASIC programs that have been saved in compressed (e.g. non-ASCII) format will now be loaded a sector at a time instead of byte by byte. This will result in an INCREDIBLE speed increase in program loads and saves.

DI, DU, and DR. DI allows you to delete a program line from one location and move it to another. DU allows you to duplicate a program line from one location to another without affecting the original line. DR performs the same function as DI, but goes through the program and changes all references to the program line being moved to reflect its new location.

Extended error posting. Errors will now be rewarded with a more complete display in which the actual statement that caused the error will be shown to the operator and indicated with an arrow. Makes locating errors in long lines MUCH easier.

OPTION statement expanded. The OPTION statement has been expanded to include two new switches; "S" and "L".

Standard Model 4 BASIC requires that you enter spaces around keywords. This is done so that you may use reserved words in long variables (and now labels, too). However, this also means that programs that have been written with the Model III may not convert easily to the Model 4.

Without the spaces, Model 4 BASIC doesn't know keywords. This allows keywords in variables, since without the spaces there is nothing special about the string of characters. We now allow you to alter this.

In the short form, reserved words themselves are once again significant with or without spaces. This is like the Model III. In the long form, of course, BASIC will react in the standard method.

This means that if a BASIC program is saved in ASCII on the Model III, transferred to the Model 4 media and loaded while BASIC is under OPTION S spaces will inserted for you. Makes converting Model III BASIC programs a snap.

INPUT@ controlled screen input. This will function only if you install the BE1 program. INPUT@ allows you to input data in a controlled manner from any location on the screen. It functions with the row, column printing and will optionally print a prompt message if you include one.

INPUT@ may be terminated by using ENTER or any of the function keys. Depending on the key pressed, INPUT@ will return different values, making it simple to take four different courses of action depending on which key was pressed.

INPUT@ replaces those INKEY$ routines for keyboard input. Much faster, more control, and better results. Saves many lines of subroutines.

Do the enhancements use up more memory? Yes, a little. And INPUT@ uses a bit more yet (that is the reason for the two separate enhancements with INPUT@ being the only difference). However, using these enhancements could save you many K of additional programming statements. And if you only lose about 400 bytes for them, in the end you come out ahead.

SORT

This utility allows you to sort up to thirty arrays of any type (string or numeric) in ascending or descending order and key on up to ten of those. You call SORT via the SYSTEM statement (or "!" abbreviation).

SORT will not affect any other areas other than the arrays indicated in the command line. It adds nothing to the size of your program. It is simply a machine language sort available to the program external from BASIC that works within BASIC.

It will enable the programmer to sort single dimension arrays in seconds. And best of all, it is upward compatible with Model III Disk BASIC's CMD"O" array sort.

The same syntax will work. You could use SR (discussed next) to replace all of the CMD"O"s with !"SORT"s.

SR

Ever want to change all of the PRINTs to LPRINTs? How about correct the spelling of a word that you used throughout the ENTIRE program in 200 different lines? Ever wish you could do that with just ONE command? No problem with SR, the global search and replace utility.

SR allows you to look for any string expression and replace it with any other string expression. This includes program text, quoted literals, or ANYTHING at all. Because SR accepts string expressions, you have the option of creating all sorts of special effects.

For example, let's assume that I have inserted prompts of asterisks throughout a program inside quotes (i.e. 10 LINE INPUT "Enter value *";A$), and I want to replace all of those with reverse video question marks (CHR$(253)). The statement would look like this:

```
!"SR","*"+CHR$(34),CHR$(253)+CHR$(34)
```

and would read like this:

> Look for all of the asterisks that are followed by a quote (to avoid program statements using asterisks) and replace them with a CHR$(253) followed by a quote.

With SR, you may also limit this to one line or a range of lines with the starting and ending line number option. So from something as simple as:

```
!"SR","PRINT","LPRINT"
```

to something as complicated as the previous example with the asterisks, SR lets you do it all. If you don't specify an replace value, it will simply search. This makes it a valuable locator. All in all, SR is the total BASIC utility.

REF

This utility is a comprehensive BASIC program cross referencer. It allows you to reference by line numbers, variables or keywords.

Optionally, you may send the output to the line printer for a hardcopy of your program reference.

Using the REF program allows you to know in an instant whether or not you have used a variable already in a program (to avoid multiple definitions), locate all references to a particular line number, or find where you've use a certain keyword.

When using the program, you can specify all variables, all line numbers, all keywords, or any combination of the above. You can also optionally reference a single line number, variable, or keyword.

Whether you are documenting variable usage in a finished program or trying to find all the places that you called the subroutine you just removed, REF will earn its keep over and over again. No BASIC toolbox should be without one.

RESOLVE

The 6.0 PLUS BASIC enhancements provide you with the ability to use label addressing. This makes software development much easier. However, you have to consider that the program you write for sale may be running on a BASIC not enhanced by 6.0 PLUS (perish the thought!). What then do you do?

Well, to remove the labels by hand could take you almost as long as it took to write the program. However, to use the RESOLVE utility takes only a few seconds.

Just call it with SYSTEM"RESOLVE" (or !"RESOLVE) and it will remove all labels and references to labels. It works in a two pass method.

On the first pass, it will resolve all references to labels into line numbers. For example, if you have a "GOTO TEST" in line 200 and the label TEST is defined in line 1000, then line 200 will read "GOTO 1000".

On the second pass, it will remove all of the NAME statements. If a NAME statement is the only item on a line, it will replace it with a remark token so that the line will still be there when it is referenced.

This also has the advantage of removing the labels, which are sort of self-documenting, from a "for sale" program. They may still get into your code, but at least they will have to work a LITTLE bit to ascertain what the various subroutines are using.

Conversely, it makes it VERY easy for the programmer writing the program or modifying the source code to know what is going on in a subroutine that is identified by a label.

Summary

The 6.0 PLUS BASIC enhancements become hard to live without after about 5 minutes of use. Whether you like them for the Model III compatibility, the increase in speed, or the advanced features, you WILL like them.

Overall conclusion

The Micro-Systems Software 6.0 PLUS series of enhancements to TRSDOS and BASIC are a genuine pleasure to use. Well documented and well written, they represent the state of the art in TRS-80 software.

These programs form a "toolbox" that will aid the programmer in software development and the novice/user in standard maintenance.

6.0 PLUS - The TRSDOS 6.0 Enhancement series

Operating System enhancement package

Introduction

Congratulations on your purchase of the Micro-Systems Software TRSDOS 6.0 enhancement package. This package consists of several very useful utilities that should aid you greatly in your disk management. The programs included are:

DISKZAP	A track sector oriented display/modify utility. Allows you to, among other things, verify disk sectors, format a single track, fill disk with data, display a sector, and modify a sector.
DISKDUMP	A file oriented display/modify utility. Allows you to proceed directly to a file's sectors without knowing its location on the disk. May be used from BASIC.
DIRCHECK	A program to verify the integrity of a disk directory. Includes the ability to repair a directory if damage is discovered.
MAP	A disk file location mapper. Allows you to discover exactly where on the disk files reside.
RESTORE	A program to restore REMOVEed files, assuming that the filename is still in the directory and the disk space hasn't been re-allocated.

These five utilities give you a powerful addition to TRSDOS 6.0 in the area that it was most sorely lacking; direct disk editing. Properly used, they can be a great asset. Misused, they can cause serious data loss. Be careful!

Installing the files on your TRSDOS

The disk these files were sent to you on contains its own TRSDOS system. To begin, place the Enhancement package Master diskette in drive 0 and press the reset button. When TRSDOS boots, respond to the "Date ?" prompt and use the COPY command to transfer the files to any of your current TRSDOS formatted disks.

Consult the TRSDOS manual under the command COPY for details in using this command.

We also strongly suggest that the first item you do, before anything else, is backup the Master disk that we sent you. We will replace Master disks that are damaged, but replacements take time. You should have several copies of the Master handy.

DISKZAP

This utility is a disk sector editor. DISKZAP can be used to display, modify, copy, or verify diskette sectors as well as format diskette tracks.

==========

DISKZAP

There are no parameters for this utility

==========

When executed, the DISKZAP utility will display its command menu on the video screen:

```
*   Set
    Fill
    Copy
    Print
    Verify
    Format
    Display
```

This is the MAIN MENU. It lists all the sub-options and allows you to move between them. DISKZAP will default to certain parameters for each drive:

40 cylinders
18 sectors on track 0
Double density track 0
18 sectors on all remaining tracks
All remaining tracks double density

Any of these may be altered via the "Set" sub-option. The asterisk that appears to the left of the "Set" option on power-up is the "control cursor". Whichever sub-option it is positioned next to is the one that will be invoked when the <ENTER> key is pressed. It may be moved up and down the list by pressing the <up arrow> and <down arrow> keys. To exit DISKZAP, from the main menu press "O" (as in "Out").

Set (Alter disk drive parameters)

DISKZAP powers up with the control cursor positioned for this sub-option. If the default parameters shown above are proper for the diskette you wish to work with, then you may proceed directly to the sub-option of your choice and begin the desired operation. If the diskette's characteristics differ from the default parameters, then you need to use the Set sub-option to alter the drive parameters.

To invoke this sub-option, as with any of the sub-options, simply position the control cursor to the left of the word "Set" and press ENTER. The first question to be asked will be:

Drivespec ?

Respond to this with the drivespec of the drive that you are configuring. After setting the drive, you will be asked:

Cylinder count ?

Answer this question with the number of cylinders on the disk that you are configuring. Enter the true cylinder count for the diskette. This parameter is interested in how many tracks are on the DISK, not how many your drive is capable of.

The next prompt is:

Surface count ?

Enter the number of data recording surfaces on the diskette in question. For a single-sided diskette, the proper value should be 1, and for a double-sided diskette, it should be 2. For rigid drives, this parameter will vary according to individual configuration.

Now DISKZAP will query:

CYL 0 sec/trk ?

This is requesting the number of sectors on track zero. The DISKZAP defaults to the proper sector count for 5-1/4" double-density diskettes, 18. If the diskette which is to be operated upon has a cylinder 0 sector count that differs from the default, enter the proper value now.

After the cylinder 0 sector count has been provided, DISKZAP will prompt with:

CYL 0 density ?

This parameter allows you to configure the density of cylinder 0 separately from the cylinder 0 sector count. Reply to this with an "S" for single density or a "D" for double density.

Note that many Model I double density system disks use a single density track zero. This is required by the ROM bootstrap loader. The data diskettes, however, format track zero as double density so that the granules not actually USED by the bootstrap can be freed to the system for data storage.

Following your definition of track 0, you will be given the opportunity to configure those same two parameters for all other tracks on the diskette.

The first query is:

Sectors/track ?

Answer this query with a value (0-255) to indicate how many sectors there are on all the remaining tracks. The standard for 5-1/4" diskettes, of course, will be 10 sectors per track in single density and 18 sectors per track in double density. However, there is the chance that some system could be using more or less sectors on the track without altering the density that the floppy disk controller works in. This parameter gives you the ability to configure for any eventuality

After configuring for the number of sectors, you will be queried as to the density of the remaining tracks. You will be asked:

Track density ?

Respond to this question with either "S" for single density or "D" for double density, depending, of course, on the density of the disk.

When using the set option, you only respond to as many prompts as are pertinent to you. For example, if all you wanted to do was change the diskette's track count, you could go to the set option and alter the track count. Then you could press BREAK and return the command mode immediately. There is no need to step through prompts that are irrelevant.

It is with this in mind that we have designed the set option. The parameters we felt you were going to use the most (cylinder count, surface count, etc.) are the first question which DISKZAP asks, such that they may be altered quickly and allow the user to avoid the rest of the prompts with the BREAK key.

Because pressing ENTER leaves the parameter unchanged instead of re-loading the original default, you do not need to re-enter a parameter that is set the way that you want it. Set will retain this drive configuration for as long as DISKZAP is in operation, but must be re-configured upon each new entry of the program.

Fill (Fill sectors with specified byte)

This option will allow the user to fill a sector with any particular byte that may be desired. This is useful when it is desired to erase completely old data from a sector without re-formatting the entire disk.

To invoke this sub-option, place the control cursor to the left of the word "Fill" in the main menu and press ENTER.

The first question to be asked is:

Drivespec ?

Reply to this with the name of the drive that contains the diskette to be operated on. Any valid drivespec will be allowed here. After answering that question, you will be asked:

Cylinder ?

Answer this question with the track number that contains the first (or only) sector to be filled. This cylinder number is entered in hexadecimal.

Once the cylinder is entered, you will be asked:

Sector ?

Reply to this query with the number of the first (or only) sector to be filled.

The next prompt will be:

Sector count ?

Respond to this with a value that represents the number of sectors, beginning with the sector specified in the preceding questions, to be filled. This cont must be entered in decimal, and may assume any value from 1 to 255. 1 is the default.

The final question will be:

Fill data ?

Answer this question with the byte that you wish to have the sector filled with. This one-byte value must be entered in hexadecimal. Pressing ENTER at this prompt will cause DISKZAP to use the default fill value, which is zero.

For example, if you wanted to fill tracks 4 and 5 of a particular double density diskette with the hexadecimal value "E5", you would answer the questions in the following manner:

```
Drive ? 0
Cylinder ? 4
Sector ? 0
Sector count ? 60
Fill data ? E5
```

After inputting all data and pressing ENTER on the last prompt, the drive will run and DISKZAP will display the track and sector number as it fills each sector.

Copy (Copy sectors)

This function will allow you to copy sectors from one disk to another or from one part of a disk to another. To invoke this command, place the control cursor to the left of the word "Copy" in the main menu and press ENTER. The first question to be asked is:

Drivespec ?

Answer this with the drivespec of the SOURCE drive. Next, you will be asked:

Cylinder ?

Answer this with the cylinder number that contains the first (or only) SOURCE SECTOR. This is the sector that is to be copied (or the first of many, whichever you desire). After answering that question, you will be asked:

Sector ?

This is prompting you for the number of the first (or only) source sector. After this is entered, you will be prompted for:

Drivespec ?

This time it is seeking the drivespec of the DESTINATION DRIVE (the drive to which you wish to copy).

The next prompt is:

Cylinder ?

Answer this with the number of the track on the destination drive that contains the first (or only) DESTINATION SECTOR.

After answering that, you will be queried:

Sector ?

This is prompting you for the number of the first (or only) destination sector. It does NOT necessarily have to be the same as the source sector (i.e. you can copy the last two sectors of track 4 on drive 0 into the first two sectors of track 7 on drive 1).

The last piece of data required will be:

Sector count ?

This prompt is seeking the number of sectors that you wish to copy. Enter the sector count in decimal.

Note that when you are using COPY, you are defining a "block" of sectors. You specify the starting point of this block on both the SOURCE and DESTINATION drives. The "sector count" prompt allows you to define the length of the block. Pressing ENTER will copy only a single sector. But, it must be a CONTIGUOUS block. You are copying sequentially from the source sector to the destination sector for the number of sectors you specify. What this means is, if you wish to copy 50 sectors, skip 200, and copy 50 more, you will have to copy each block of 50 separately. You may, if you wish, locate them beside each other on the destination drive, but they must be copied independently.

For example, if you wished to copy track 2, sector 5 of drive 0 into track 3, sector 12 of drive 1, you would answer the prompts in the following manner:

```
Drivespec ? 0
Cylinder ? 2
Sector ? 5
Drivespec ? 1
Cylinder ? 3
Sector ? 12
Sector count ? 1
```

If you wished to copy an entire Model 4 TRSDOS 6.0, 40 track double-density diskette from drive 0 to drive 1, you would answer the prompts in the following manner:

```
Drivespec ? 0
Cylinder ? 0
Sector ? 0
Drivespec ? 1
Cylinder ? 0
Sector ? 0
Sector count ? 720
```

After answering the "sector count" query and pressing ENTER, DISKZAP will begin the copy. When copying sectors, DISKZAP will seek to read in as many sectors as it can (up to one complete track) before writing them, as opposed to reading and writing a single sector at a time.

When copying a single sector, there will be no operational difference. However, when copying more than a track (especially an entire disk), it makes LARGE difference. DISKZAP will also displays the track and sector number of each sector as it is copied (both the SOURCE sector as it is read and the DESTINATION sector as it is written).

If DISKZAP encounters an error during the sector copy routine, it will pause and display the error discovered. It will also ask if you wish to continue. It would then write as much of the source sector as it could read into the proper destination sector and proceed from there. This will allow you to copy as much data as is absolutely possible from a disk without having to work around known bad sectors. This "proceed after error" feature becomes a key one in repairing blown diskettes. If you can copy a complete track save one sector, then you have only lost 256 bytes of data as opposed to potentially much more.

Print (Print hardcopy of selected sectors)

This command will create printed copy of the contents of specified sectors. To invoke this option, position the control cursor to the left of the word "Print" in the main menu and press ENTER.

The first question asked will be:

```
Drivespec ?
```

Answer this with the drivespec of the drive that contains the first sector to be printed. Next you will be asked:

```
Cylinder ?
```

Answer with the number of the cylinder that contains the first (or only) sector to be printed. Following that, you will be queried:

```
Sector ?
```

Enter the number of the first (or only) SECTOR TO BE PRINTED. Finally, you will be prompted:

Sector count ?

Reply to this with the number of sectors that you wish to print. Remember, just as with copy, you are dealing with contiguous blocks ONLY! You may not print 5 sectors on track 0 and then 5 on track 11 without printing them both independently of one another.

For instance, in order to print out all the directory sectors (assuming the directory was on track 11 hex) from the double density diskette in drive 0, you would:

Drivespec ? 0
Cylinder ? 11
Sector ? 0
Sector count ? 18

As each sector is printed, it will be displayed on the screen. You may tell by examining the track and sector indicators in the upper left hand corner of the screen which sector is currently being printed.

Please note that DISKZAP does NOT check for printer ready status. If you engage the print option and there is no printer available, DISKZAP will simply "lock up" and force you to either make a printer available or reset the machine.

Verify (Read and check specified sectors)

This option will allow you to read and verify any specified sectors on the disk. It will check each sector for accuracy by verifying the CRC byte. If it encounters an error, it will pause with the correct error message. Pressing ENTER will cause it to continue verifying.

To invoke this option, as with any other, position the control cursor to the left of the word "Verify" and press ENTER. The first question asked will be:

Drivespec ?

Reply to this question with the drivespec of the drive that contains the diskette that you wish to verify. The next question asked will be:

Cylinder ?

This is prompting you for the cylinder number that contains the sector you wish to begin verifying at. When verifying an entire diskette, you may press ENTER at this prompt to select track 0. After answering that, you will be asked:

Sector ?

This is asking you for the sector number on the above specified track that you wish to begin verifying at. This would allow you to begin verifying with the last two sectors of track 5. Following that, you will be prompted:

Sector count ?

This is seeking the number of sectors, in decimal, you wish to verify. Remember, if you specify more sectors for a disk than you have configured for in "Set", it will wrap around from the last configured track and begin again at track 0, sector 0 and continue from there. That is why it's important to configure for the correct track count before beginning with any diskette.

The final query that DISKZAP will ask is:

Ignore data AM ?

This question requires a yes/no answer. When answered with a "Y", DISKZAP will not display a message to inform the operator that a special type of data address mark, which is reserved for use by the directory, has been detected. If the question is answered with an "N", DISKZAP will print the following message and pause until a key is depressed whenever the special address mark is encountered:

AM/WRITE FAULT

While you are verifying a diskette, you may abort and return to the main menu by holding down the BREAK key.

As an example, suppose it were desired to verify a 35-track, single-density diskette in drive 1. The following data would be provided to the Verify command:

Drive ? 1
Track ? 0
Sector ? 0
Sector count ? 350

Once you have answered the final question and pressed ENTER, DISKZAP will begin reading the specified sectors. It will display the track and sector number as it verifies each sector. As each sector is read, the CRC value is calculated and checked and any errors reported.

Format (Format a selected track or tracks)

This sub-option allows you to format a track or series of tracks. You may, if you wish, use it to reformat a track somewhere in the middle of a disk to repair a non-readable sector. To invoke this option, position the control cursor to the left of the word "Format" in the main menu and press ENTER.

The first question is:

Drivespec ?

This is prompting you for the drivespec of the drive that contains the disk you wish to format a track on. After answering that, you will be queried:

Cylinder ?

Respond to this with the number of the cylinder at which you wish to begin formatting. The next question is:

Cylinder count ?

This is seeking the information as to how many cylinders you desire to format. Pressing ENTER at this prompt will default to one track.

The final question under the Format command is:

Interleave factor ?

The interleave factor determines the order in which sectors are numbered on the diskette, and can have a profound effect on disk access speed. The normal values for sector interleave factor are 2 for 5-1/4" single-density diskettes, and 3 for 5-1/4" double-density diskettes.

Please note that the DISKZAP Format command is not interchangeable with the TRSDOS utility FORMAT. The DISKZAP Format command is simply capable of performing cylinder formatting; the FORMAT utility not only formats a diskette but also initializes the diskette with a great deal of system information, including a bootstrap and disk directory.

Display (Display or modify diskette sectors)

This is perhaps the most often used option in DISKZAP, and the heart of the disk editor. DISKZAP uses a full screen editor that has cursor wraparound.

To invoke this sub-option, position the control cursor to the left of the word "Display" in the main menu and press ENTER.

The first question you will be asked is:

Drivespec ?

Answer this query with the drivespec of the drive that contains the diskette with the sector you wish to display/modify. The next question is:

Cylinder ?

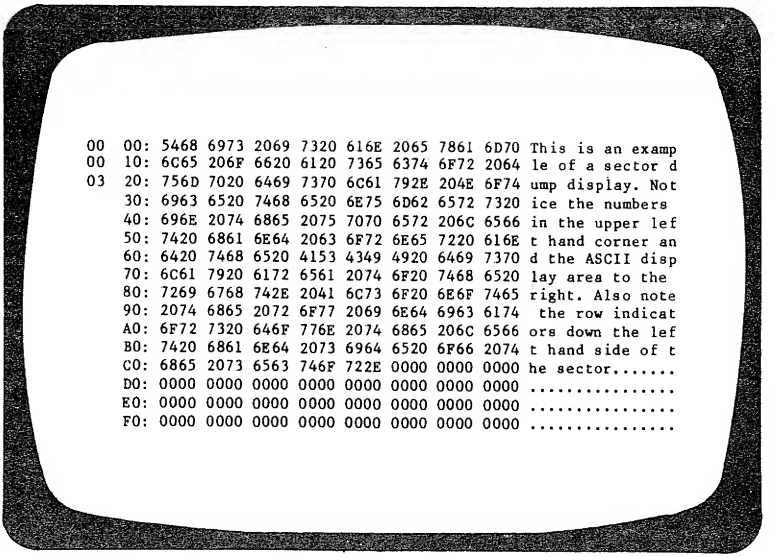
This is prompting you for the number of the cylinder on the disk that contains the sector you wish to examine.

The final question is:

Sector ?

Reply to this with the number of the sector you wish to display.

After typing in the sector number and pressing ENTER, you should see a display that looks something like this:



At the upper left-hand corner is a one-byte hexadecimal value which relates the logical device number of the disk drive currently addressed. Note that this is a device number and not a drive specification. The number listed immediately below this is the current cylinder number, and below that is the current sector number. The column of digits slightly indented from the left margin are the BEGINNING BYTE INDICATORS. Each one of those indicates the number of the first byte in that row. Then there are rows of 16 bytes each (10 hex). This is the HEXADECIMAL DISPLAY AREA. These are set in groupings of two bytes, such that you have eight columns of two separated by spaces. Immediately to the right of the hexadecimal portion of the sector display is the ASCII DISPLAY AREA. There are 16 ASCII characters on a row corresponding to the bytes in the hexadecimal display row immediately to its left. Non-ASCII characters will be displayed as periods.

At this point, you have several options, each of which is controlled by a single keystroke. They are:

Key	Function
;	Increment display position one sector
+	Increment display position one cylinder
-	Decrement display position one sector
=	Decrement display position one cylinder
BREAK	Return to main menu
M	Enter modify mode

If you select "M" to enter the modify mode, the display will change slightly and you will have several other options.

If DISKZAP encounters an error during a sector read in the display mode, it will pause and display the error discovered. It will also ask you if you wish to continue. If you respond "Y", it will display as much of the sector as it could read. You may then enter the modify mode and make any corrections possible before re-writing it. The sector will be re-written to the disk reflecting any corrections you may have made. That means there will no longer be a read error from system level. It does not mean that the data is now 100% correct. It is correct only to the level that were able to repair it, but it will read as it is now without an error. This "continue after error" feature will allow you to rescue bad sectors in part or in whole, where otherwise you would have had no chance of recovering the data.

When you enter the modify mode, a reverse video cursor will appear over the byte in the upper left hand corner. You move the cursor about within the sector by using the arrow keys. Whatever byte is the graphic block cursor is currently positioned over is referred to as the CURRENT CURSOR LOCATION. This is the byte that will be affected should you enter a change.

Upon entering the modify mode, two additional pieces of information will be displayed in the upper left hand area of the screen. Immediately underneath the current sector number will appear the current cursor location. This will change as you move the cursor about in the sector. Underneath that will be the value of the byte at the current cursor location.

At this point, you have several options, each of which is controlled by a single keystroke. They are:

Key	Function
right arrow	Increment cursor position one byte
down arrow	Increment cursor position one row
left arrow	Decrement cursor position one byte
up arrow	Decrement cursor position one row
BREAK	Aborts modify mode and returns you to the main menu without re-writing the sector. Restores original contents.
ENTER	Terminate modification mode and returns you to the display mode after writing the modified sector to the disk.
F1	Toggles sector display between hexadecimal and ASCII character display mode
@	Fills entire sector, starting at current cursor position, with "00" bytes.
@ xx	Fills "xx" bytes, starting from the current cursor position, with "00" bytes.
@ xxyy	Fills "yy" bytes within the current sector from current cursor position with data byte "yy"

To modify a byte, position the cursor over the proper byte and enter the two-digit hexadecimal value (if in the hex display mode) or single-character value (if in the ASCII character display mode) which the byte is to be changed to. When you finish modifying one byte, the cursor will move onto the next. If that was that last byte of a row, the cursor will move onto the first byte of the NEXT row. The only exception is the last byte of the last row. After modifying it, the cursor will stay right where it is. To begin with the next sector, write this one back to the disk with ENTER, advance to the next sector with ";", enter the modify mode again with "M", and return to modifying.

NOTE: As a general rule, DISKZAP expects all cylinder and sector addresses as well as fill data to be entered in hexadecimal format. Cylinder and sector counts, on the other hand, are assumed to be entered in decimal.

DISKDUMP

This is a machine-language disk sector display/modify utility.

DISKDUMP [filespec]

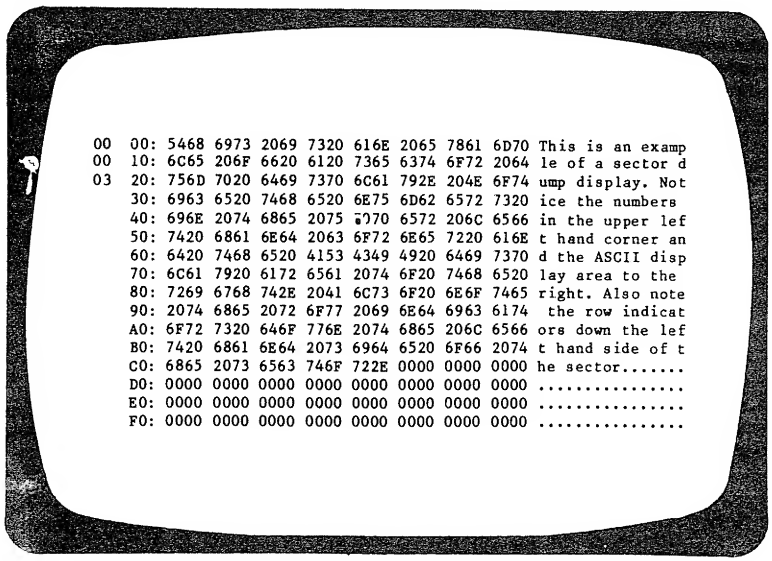
"[filespec]" is the optional name of the file to be examined/modified.

If no filespec is given on the DISKDUMP command line, the program will prompt for a filename by displaying the asterisk prompt.

Enter the file name and include all extensions and passwords, if any. DISKDUMP will now display the first sector of the file. The following commands are available:

Key	Function
;	Advance one sector
-	Go back one sector
+	Advance to end of file
=	Go to beginning of file
F	Find hexadecimal value
G	Go to specified sector
M	Enter modify mode
P	Locate address in load module file

A sector display will look like this :



The two-digit number in the upper left-hand corner of the screen indicates the disk drive device number which the file is resident upon. Note that this is not the disk drive specification, or name; it is the drive device number, and may have a value of 0-7.

Immediately below the drive device number is the number of the physical record currently displayed.

Slightly indented from the left side of the display is a column of two-digit hexadecimal numbers. These numbers indicate the relative byte number of the first byte displayed on each line. For example, in the DISKDUMP display above, the fifth row down from the top begins with the number 40. This means that the first byte on that line is relative byte 40H, the next is relative byte 41H, the eleventh byte is 4AH, etc.

To the right of the row numbers is the actual information contained within the physical record itself. This information is displayed in hexadecimal by default, in eight groups of two bytes each. This hexadecimal display may be exchanged for an ASCII character display by pressing the <F1> key on the TRS-80 keyboard.

Either the next or the previous physical record in the file may be displayed by pressing the semicolon, ";", key (for the next record) or the minus, "-", key (for the previous record. If an attempt is made to display a record outside the limits of the file, the command will be ignored.

The first record or the last record in the file may be displayed at any time by pressing the equal, "=", key (for the first record) or the plus, "+", key (for the last record).

DISKDUMP may display any given record if the "G" command is used. Simply press the "G" key, followed by the desired record number, in hexadecimal. After pressing <ENTER>, DISKDUMP will display the proper record.

DISKDUMP's "F" command is used to find any occurrences of any given hexadecimal value within a physical record. In order to find the occurrences of a byte, type "G" followed by a two-digit hexadecimal value (if in the hexadecimal display mode) or a single ASCII character (if in the ASCII display mode), and press <ENTER>. DISKDUMP will now flash a graphics block in the position occupied by any matching bytes within the record. To abort the "find" display, press any key.

Physical records may be edited by entering DISKDUMP's modify mode. To enter the modify mode, press the "M" key from the record display mode. A reverse video cursor will appear in the upper left-hand corner of the record display. This cursor may be moved about the sector display at will with the four arrow keys on the TRS-80 keyboard. When the cursor is positioned over a byte that is to be modified, simply enter the two-digit hex value (in the hexadecimal display mode) or the single ASCII character (if in the ASCII display mode) which the byte is to be changed to. The cursor will automatically advance to the next byte within the record, moving to the next line of the record display if necessary. When all changes are complete, press the <ENTER> key to write the updated record to diskette. If it is not desired to save the changes to diskette, the <BREAK> key may be depressed to cancel all changes.

When you enter the modify mode via the "M" command, you will have the following options:

Key	Function
up arrow	Decrement cursor position one row
down arrow	Increment cursor position one row
right arrow	Increment cursor position one byte
left arrow	Decrement cursor position one byte
sh up arrow	Home cursor
@ xx	Fill "xx" bytes with 00 value, starting at current cursor position
@ xxyy	Fill "yy" bytes with "xx" byte, starting at current cursor position

When editing program files save in load-module format, the "P" command can be extremely useful. By typing "P" followed by a hexadecimal address will cause DISKDUMP to search the file for the byte which will load into the specified address in RAM. When DISKDUMP finds the byte, it will automatically enter the modify mode and position the cursor on the proper byte. If DISKDUMP is unable to locate the address, the sector display will be cleared and the message "Invalid data" will be displayed on the screen. DISKDUMP will then return to the filename prompt.

The "@" command is used to fill a record or a portion of a record with a user-defined byte. The simplest form of this command is:

```
@ xx
```

where "xx" is a two-digit hexadecimal value which defines the number of bytes that are to be filled (starting at the current cursor location) with a 00 byte. A slightly more complex form of this command is:

```
@ xxyy
```

where "yy" is a two-digit hexadecimal value which defines the number of bytes to be filled (starting at the current cursor position) with the byte defined by "xx".

DIRCHECK

This utility is used to check the integrity of a diskette's file directory, and optionally, to repair certain types of damage to the directory.

==========

DIRCHECK [:dr] [TO] file/*device (param=exp)

":dr" is the drive specification of the disk drive containing the diskette whose directory is to be examined

"file/*device" is the optional output file or device to which all messages concerning the state of the diskette's directory are routed.

The allowable parameters for DIRCHECK are:

FILES	Instructs DIRCHECK to repair faulty file directory entries, if any
GAT	Instructs DIRCHECK to repair a faulty Granule Allocation Table, if necessary
HIT	Instructs DIRCHECK to repair a faulty Hash Index Table, if necessary

Abbreviations:

FILES	F
GAT	G
HIT	H

==========

The DIRCHECK utility may be used to automatically examine a diskette directory and report any errors or inconsistencies within the directory. The simplest form of the DIRCHECK command is:

DIRCHECK :dr

where ":dr" is the drive specification of a disk drive containing the diskette whose directory is to be examined. DIRCHECK will read the diskette directory and display a list of any errors found on the video display. After the list of errors, if any, DIRCHECK will print "DIRCHECK complete, xxx total errors", where "xxx" is the number of directory errors found by the DIRCHECK program.

Also note that if you do not specify any options upon entering DIRCHECK, it will prompt you with an asterisk. You may at that time enter all needed options. Pressing BREAK will return you to DOS.

The list of errors may also be directed to any other character-oriented device, or to a file, by specifying an optional output channel. For example, the DIRCHECK command:

```
DIRCHECK :2 TO *PR
```

will output the list of directory errors to the lineprinter.

```
DIRCHECK :0 ERRORS/TXT
```

will output the list of directory errors to the file ERRORS/TXT. Note that the TO was omitted in this example, it is optional.

DIRCHECK is also capable of repairing certain directory errors. The parameters FILES, GAT,and HIT are used to inform DIRCHECK of which portion(s) of the diskette directory should be repaired. If the FILES parameter is specified, DIRCHECK will repair any errors in the file entry table of the diskette directory. Likewise, if the GAT or HIT parameters are provided, DIRCHECK will fix any errors encountered in the respective table. If any of the parameters are omitted, DIRCHECK will report, but will not repair, any errors discovered in the respective area of the directory.

To specify one of the fix options, include the parameter in parenthesis after specifying the drive number or, if it exists, the optional output file/device. Therefore, to instruct DIRCHECK to repair the GAT, you would use the following syntax:

```
DIRCHECK :2 (GAT)
```

if an output field is specified:

```
DIRCHECK :2 TO *PR (GAT)
```

Note that DIRCHECK (or any "directory fixing" program) is incapable of repairing certain directory discrepancies, listed below:

Error	Possible cure
Locked gran assigned to file	Kill offending file
Granule multiply assigned	Kill offending file
Granule assigned past cyl count	Kill offending file
BOOT/SYS not found	Restore BOOT/SYS file
BOOT/SYS not assigned space	Restore BOOT/SYS file
DIR/SYS not found	Restore DIR/SYS file
DIR/SYS not assigned space	Restore DIR/SYS file

Also note that although many directory errors can be repaired by DIRCHECK, it is possible that certain files whose directory information was in error may be adversely affected. In other words, if a directory has bad information in it and DIRCHECK repairs it to the best of its ability, it may cause one file to have its data area lost in order to preserve the integrity of the entire directory. Sometimes the errors are simply too severe to correct without unwelcome side effects.

MAP

The MAP utility provides a list, by cylinder and sector, of the areas allocated to files on a diskette.

==========

MAP [:dr] [TO] file/*device (param=exp)

":dr" is a disk drive specification

"file/*device" is a file or character-oriented device to which the output from MAP will to sent

The allowable parameters are:

SYSTEM	Include system files in MAP listing
INVIS	Include invisible files in MAP listing
HEX	Provide cylinder & sector numbers in hexadecimal

Abbreviations:

SYSTEM	S
INVIS	I
HEX	H

==========

The MAP command may provide a file-by-file list of diskette space allocation. To display all files on a diskette, and the areas occupied by them, type:

```
MAP :dr
```

where ":dr" is the drive number which you wish to MAP. The screen will display something like the following:

```
[TRSDOS 01/26/83]
MAILLIST   001,006 - 001,011
CLICK/FLT  001,012 - 001,017
BASIC/CMD  003,000 - 003,005
```

Each filename is followed by a set of numbers. Take, for example, the case of MAILLIST, in the listing above. The MAP tells us that MAILLLIST begins on cylinder 1, sector 6 and continues through cylinder 1, sector 11. Large files may contain more than one segment of this sort. For instance, in the MAP shown below, the file FDAT is divided into five separate segments:

```
[Utility  01/31/83]
FDAT        012,000 - 017,005
            018,012 - 019,017
            021,000 - 025,017
            027,012 - 029,017
            034,000 - 036,011
```

Note that the MAP utility can provide information in hexadecimal notation as well as decimal, if the HEX parameter is specified. The fist example, above, would appear like this in hexadecimal format:

```
[TRSDOS  01/26/83]
MAILLIST    01,06 - 01,0B
CLICK/FLT   01,0C - 01,11
BASIC/CMD   03,00 - 03,05
```

The SYSTEM and INVIS parameters may be used to cause the MAP utility to display a MAP of system files and invisible files, respectively. Without these two parameters, only the visible user files will be displayed. You may abbreviate these parameters as "S" and "I".

Output from MAP is normally sent to the video display, but it may be re-directed to any other device, or to a file. For instance, to obtain a printout of a MAP of all files on drive 2, type:

```
MAP :2 *PR (SYSTEM,INVIS)
```

to send it to a file:

```
MAP :2 TO MAP/TXT:1 (S,I)
```

Note that the TO preceding the output file/device is optional.

If you do not specify any options on the command line, MAP will load and prompt you with an asterisk. You may enter the proper options then.

RESTORE

The RESTORE utility is used to reclaim files which have been REMOVEed.

RESTORE [filespec]

"filespec" is the name of a REMOVEed file.

Note that RESTORE cannot reclaim any REMOVEed file. Certain conditions must be met:

(1) The disk space originally allocated to the file must not have been reassigned to another file.

(2) The primary and any extended directory entries for the file must not have been altered in any way.

If either condition is not met, RESTORE will issue the message "Disk space has been re-allocated", and will abort.

If you do not specify the filespec from the command line, RESTORE will prompt you with an asterisk and you may enter the filespec at that time.

When attempting to recover a file, RESTORE will search all available drives (unless a drive specification is explicitly provided in the command line) for the file. If RESTORE locates an active file bearing the same name, it will abort with an error. If this occurs, and the second file is NOT the same as the file you want to recover (e.g. TEST that has been REMOVEed on drive 1 and TEST that is active on drive 2), simply specify TEST:1 in the RESTORE command line. If RESTORE is able to recover the file, the utility will exit to DOS, and the RESTOREd file will be immediately useable.

NOTE: When attempting to recover a file, RESTORE will reclaim the first occurrance of the proper filename in a diskette directory. If the same filename has been created and REMOVEed several times upon a diskette, RESTORE may not recover the same occurrance of the file as was intended. If this is the case, the improper file should be RENAMEd and REMOVEed. The RESTORE process may then be repeated until the proper file is recovered.

BASIC interpreter enhancement package

Introduction

Congratulations on your purchase of Micro-Systems Software's Enhancements to TRSDOS 6.0's BASIC. We hope that you will be pleased with your purchase and that it will make your usage of BASIC far more profitable. This package consists of the following files:

SR	Global text editor
REF	BASIC cross referencer
RESOLVE	Label resolver
SORT	BASIC array sort
BE1	General enhancements #1
BE2	General enhancements #2

Installing your enhancements

Unlike the DOS enhancements that are simply programs on the disk to be executed, the BASIC enhancements must be "installed". Which is to say that you must first alter the BASIC/CMD file itself. This is done via the library commands LOAD and DUMP. First, however, you must decide which of the two general enhancement programs you wish to use (i.e. BE1/CMD or BE2/CMD).

As you know, any sort of enhancements to BASIC are going to use some memory. We have tried to keep the amount of memory used by the enhancements to a minimum, going in many cases to an external utility. However, you can count on the fact that BASIC enhanced by 6.0 PLUS will not have quite as much free memory as the standard Model 4 BASIC.

However, it is our belief that you will discover programming with the 6.0 PLUS BASIC Enhancements will save you much more memory in eliminated program statements than it will ever cost you to install.

However, to further aid you in your selection, we have included TWO versions of the general BASIC enhancements, BE1 and BE2. The program BE1 includes the INPUT@ controlled screen input routine and, of course, BE2 does not. This represents to you savings of about 200 bytes of memory. It is our suggestion that you read the documentation where it discusses INPUT@ and make the decision, based on that information, whether or not you wish to spend the additional memory.

The specifics

Place a backup copy of the 6.0 PLUS series Master diskette in drive 0 and reset the machine. At the "TRSDOS Ready" prompt, enter the following:

```
MEMORY (CLEAR)
LOAD BASIC/CMD.BASIC
```

Now, decide which edition of the enhancement you will use. If it is BE1, enter the commands:

```
LOAD BE1/CMD
DUMP BASIC/CMD.BASIC (START=X'3000',END=X'864F',TRA=X'8407')
```

If you decide on BE2 (no INPUT@), use the following commands:

```
LOAD BE2/CMD
DUMP BASIC/CMD.BASIC (START=X'3000',END=X'8531',TRA=X'82E9')
```

This will create a copy of BASIC on the disk that contains the enhancements. This copy of BASIC may be overlaid onto existing copies if you use the password "BASIC" when entering the COPY command.

Once they're installed

Remember, you cannot use any of the other enhancements unless you install one of the two general enhancement packages. Also remember that the ONLY difference between the two is the presence or absence of INPUT@.

When you enter BASIC now, the screen will not clear. In addition to the standard MicroSoft and Tandy copyright notice, you will see a notice to the effect that the enhancements are copyrighted by Micro-Systems Software Inc.

Finally, the current amount of free memory and the number of file buffers allocated will be displayed as part of the header.

BASIC is not entered or operated in any new fashion. There are no steps to be re-learned. The 6.0 PLUS Enhancement series is invisible to the user until they implement one of the extended commands.

DI (delete and insert BASIC program line)

This command will allow you to remove a BASIC program line from one location and insert it in another.

==========

DI pln,nln

pln is the present line number.

nln is the new line number.

==========

This command is used to delete a line number in a BASIC program and insert that line into the program at another point.

Example

DI 100,122

This will copy line 100 to line 122 and then delete line 100.

DU (duplicate BASIC program line)

This command will duplicate a BASIC program line from one location to another.

==========

DU pln,nln

pln is the present line number.

nln is the new line number.

==========

This command is used to copy a line number in a BASIC program to another point in the program. The line will now exist at both the old and the new point.

Example

DU 100,122

This will copy line 100 to line 122 and still preserve line 100.

DR (delete and insert BASIC program line with renumber option)

This command performs the same function as DI, but it will alter all references to the line to reflect its new location.

==========

DR pln,nln

pln is the present line number

nln is the new line number

==========

This command is used to copy a BASIC program line from one point in the program to another. It will delete the line at its old location. In that respect, it functions in the same manner as the DI command. However, DR will also renumber any references to the line moved so that program flow is not interrupted.

What this means is that if you have a line 100 that states "GOTO 1000" and you use DR to move line 1000 to line 1050, line 100 will be altered to refer "GOTO 1050". In cases where this is not desired, use DI. By including both commands, we hope to cover all situations.

Example

```
DR 100,122
```

This will copy line 100 to line 122 and then delete line 100. After that, it will alter any and all references to line 100 to reflect its move to line 122.

SHORTHAND

Our BASIC enhancements include some highly convenient shorthand for most of your commonly used commands and statements. They are divided into two areas: immediate commands and abbreviated statements.

Immediate commands:

Command	Function
up arrow	List preceding line of program
down arrow	List next line of program
shift up arrow	List first line of program
shift down arrow	List last line of program
; (Semi colon)	List first line of program
/ (Slash mark)	List last line of program
. (Period)	List current line of program
, (Comma)	Edit current line of program.

Abbreviated statements:

Abbreviation	Statement
A	AUTO
D	DELETE
E	EDIT
G	GOTO
I	INPUT
K"	KILL
L	LIST
L"	LOAD
N	NAME
R or R"	RUN
S"	SAVE
!	SYSTEM

Any of the immediate commands must be the first character typed for that line. In other words, if you already typed a character and backspaced, you should still press ENTER to get a "fresh" command line before using shorthand commands.

Abbreviated statements (L,E,D,L",S,etc...) may appear anywhere within a program line. When BASIC encounters them, it will expand them to their normal state.

There are certain conditions that will disable shorthand commands. If JCL is active, shorthand commands are disallowed. Also, if you have implemented the "protected program" option, shorthand will not function.

When using shorthand for abbreviated statements, it is no necessary that you put spaces around them as you are typing. When they are expanded, the spaces will be inserted for you.

REF (reference BASIC program text)

This command gives you a comprehensive BASIC program cross referencer.

==========

SYSTEM"REF",option,option...

option can be any of the parameters legal for this command.

Parameter	Function
S	Single variable, line, or keyword
V	All variables
L	All line numbers
K	All keywords
P	Printer output

==========

This will allow you to reference your BASIC program for line numbers (L), variables (V), or keywords (K). For example:

```
SYSTEM"REF",K,L,V
```

This will reference the program for all three. If you specify a P also (SYSTEM"REF",K,L,V,P), REF will do the same thing, but it will output it to the line printer.

To display a single variable you use the "S" parameter. For example:

```
SYSTEM"REF",S,A
```

Every time the variable "A" occurs in the text will be listed for you. It becomes as specific as you are. For example:

```
SYSTEM"REF",S,A$
```

will only hunt up references to the variable "A" when it is being used as a string variable. And still further:

SYSTEM"REF",S,A$(

will look up only references to "A" as an ARRAY string variable. It will also take complex variable names like:

SYSTEM"REF",S,FINDIT

This will look for all occurrences of the variable "FINDIT". The same syntax applies for a single line number or a single keyword. For example:

SYSTEM"REF",S,PRINT

will reference all the PRINT statements.

Please note that when the "S" parameter is specified, the only other information that may appear in the options list is the item being referenced. For example:

SYSTEM"REF",S,FNAME$,L

will not reference "FNAME$" and then all line numbers. It will produce an illegal function call error. You may, however, include the "P" parameter to get hardcopy of your reference. For example:

SYSTEM"REF",S,FNAME$,P

is legal.

Please note that the "AS" keyword in FIELD statements is regarded by BASIC as a variable. Therefore, we also see it as a variable. Do not let this surprise you when it happens. It is simply an "oddity" of BASIC.

SR (global editing of BASIC text)

This command allows you to search for and display or replace any ASCII string literal or expression that occurs in BASIC text.

==========

SYSTEM"SR",sexp,rexp,sln-eln

sexp is the search expression. This can be any valid string expression, a literal, or a combination of both string variables AND literals.

rexp is the replace expression. This can also be any valid string expression, literal, or combination of both.

sln-eln are the optional starting and ending line number. This allows you to restrict your editing to one block of text. If not present, it will be a global edit of the entire text. If you specify "sln" only, it will do only that line number. If you specify "sln-", it will begin at that line number and go to the end of text.

==========

This is a great programmer's tool. It will search for and display or replace any string variable or expression. To engage it, type SYSTEM"SR" (for search and replace) followed by a literal ASCII string, OR any character string or other string variable. After it alters a line, it will list that line showing the change. It operates in two modes: Search mode and Search and Replace mode. For example, if you type:

SYSTEM"SR","Test"

It will look through the text and list every line with the word "Test" in it. If you type:

SYSTEM"SR","Test","NewTest"

It will look through the text and every time that it finds the word "Test", it will replace it with the word "NewTest". If you type:

SYSTEM"SR","Test","NewTest",100-200

It will confine this procedure to lines 100 through 200. You can also use a combination of variables and literals. For example:

SYSTEM"SR",":",CHR$(10)+":"

This will go through the whole text and insert a line feed in front of every colon.

NOTE: Because BASIC will not allow control characters to be imbedded in strings and printed, be certain that you do NOT use SR to replace characters within a string literal with control characters. To print control characters, you must use the CHR$(x) function of BASIC. You may, if you wish, use a single character control sequence and use SR to replace all of those with the proper CHR$(x) command.

SORT (sort BASIC arrays)

This command allows you to sort BASIC arrays of any type in ascending or descending order. This utility is upward compatible with the CMD"O" array sort in Model III Disk BASIC.

==========

SYSTEM"SORT",exp,[+ or -]AN$(se)+KA$-KA%,TA#,TA!

exp expression to indicate number of elements to be sorted (integer).

+ or - indicates primary key array to be sorted in ascending or descending order. Optional, if omitted ascending order will be assumed.

AN$(se) primary key array. Subscript indicates starting element number.

KA$ next key array. Plus (+) indicates ascending order.

KA% next key array. Minus (-) indicates descending order.

TA# First tag array.

TA! Next tag array.

==========

A "key" array is defined as being an array that SORT will consider when sorting. A "tag" array, on the other hand, is simply "along for the ride". When SORT finds two elements of a key array that need to be swapped, it will swap the corresponding elements of all other key arrays and all tag arrays.

You may up to ten key arrays, counting the primary array, and up to twenty tag arrays for a total of up to thirty arrays.

You must completely define all "KEY" arrays prior to defining "TAG" arrays. Please note that all key arrays are preceded with a plus (+) or a minus (-) to indicate ascending or descending order. Do not use commas. After you append the first array with a comma, SORT will assume that you are beginning the tag arrays and will consider no more key arrays.

Differing from this is the PRIMARY KEY ARRAY. The primary key array is separated from the element count by a comma for TRSDOS compatibility. If you wish descending order, you may insert an optional minus (-) between the comma and the primary key array name. A plus (+) is also legal but not needed as ascending order is assumed.

When SORT is ordering the arrays, you may interchange string and numeric arrays as you have need. In the example above, we have attempted to illustrate this. You may sort strings only or numbers only if you wish, but the option to mix them together is open to you.

Also, when SORT is ordering the arrays, if it finds a discrepancy in one of the key arrays, it will swap the elements and stop there. In other words, if the primary key array needs to be swapped, it will not even look at any of the other elements. Only when two array elements are the same will a secondary key array be considered for the sort.

Examples

```
SYSTEM"SORT",100,A$(1)+B$-C$,D$,E$,F$
```

This command line would instruct SORT to sort 100 elements of string array beginning with the first element in the array "A$". If it finds a match there, it will attempt to sort by the corresponding two elements in "B$". If it finds a match there, it will sort by the corresponding two elements in "C$". However, "C$" is sorted in descending order. Any time that it swaps an element in any of the key arrays, it swaps it in all the other key arrays and then it also swaps the corresponding elements in "D$", "E$", and "F$" (although the order of these is not important).

The "corresponding element" is defined as being those elements with the same position number. For example, the corresponding elements in the above example would be:

```
* A$(1) - B$(1) - C$(1) - D$(1) - E$(1) - F$(1)
* A$(9) - B$(9) - C$(9) - D$(9) - E$(9) - F$(9)
```

This sort is also capable of sorting integer, single precision, and double precision arrays. You may mix and match arrays. For example, to return to the sort command above, you could make "C$", "C#" with no problem. The syntax is identical.

```
SYSTEM"SORT",N%,A$(1)
```

This will function exactly the same as the old Model III Disk BASIC sort. It will sort "A$" in ascending order starting at element one and proceeding for "N%" elements.

Technical notes: Please note that the arrays may ONLY be single dimension and you may not specify a starting element number for any array other than the primary key array.

Sample use

This sample program will create a sorted index for a mailing list.

```
5 CLEAR 2000 : CLS
10 OPEN"R",1,"MAIL/DAT",52
20 FIELD 1,10 AS DUMMY$,20 AS FNME$
30 EF=LOF(1) : DIM A$(EF),RN%(EF)
40 FOR I=1 TO EF
50 GET 1,I
60 A$(I)=FNME$ : RN%(I)=LOC(1) : NEXT I
70 CLOSE
80 SYSTEM"SORT",EF,A$(1),RN%
90 OPEN"R",1,"MAIL/INX",2
100 FIELD 1,2 AS NR%
110 FOR I=1 TO EF
120 LSET NR%=MKI$(RN%(I)) : PUT 1,I : NEXT I
130 CLOSE
```

After this, whenever you want an alphabetical listing of your file, simply open the file "MAIL/INX". Those two byte records contain integer record numbers. Get each record in turn and then get the data record that it points to. Print that data and you will have an alphabetical listing.

Finally, a couple of short notes. If you have used OPTION BASE to alter the base array starting element, this is supported. In other words, if you have set the base starting element for 1, attempting to sort an element 0 will cause an error. Also, please note that long variable names are allowed within SORT.

INPUT@ (controlled screen input)

This command allows you to input string data from anywhere on the screen with control of format and character entry.

==========

INPUT@<pos>,"prompt",fl,it;var$

<pos> is the screen position you wish to input at. It will begin here with the prompt string if one is specified.

"prompt" is the prompt message you wish to have displayed on the screen in front of your input field. This must be a literal.

fl is an integer expression that defines field length.

it is the item type flag. Should be "$" for alphanumeric or "#" for numeric. If you append an asterisk to this, you set the "return on full field" mode. This may be a literal OR an expression.

var$ string variable that data typed into the input field is passed to BASIC in. Must be a string even if input was restricted to numeric only. Note that this option is separated by a semi colon. This is NOT an option. A comma will not work in that location.

==========

This utility will serve to replace many of the tiresome INKEY$ subroutines that you now have to use. Your current routines (using INKEY$) are being slowed down by BASIC's string handling functions. That fact that you are collecting data via a subroutine that handles strings and is interpreted in BASIC results in very slow keyboard response. INPUT@ will banish these problems.

Although INPUT@ does only limited error checking of itself, it does allow you to do as complex an error check as you wish later.

Your parameters are:

<pos> (Screen position). This can be anywhere from 0 to 1919 (or you may optionally use the "row,col" format). It is the same as a PRINT@ location. Whatever this value is, that is the location that INPUT@ will print the prompt string. If no prompt string was defined, then INPUT@ will put the input field start at that location.

"prompt" (Prompt message). This must be a literal. It will be printed at the location specified by <pos>. If this is not specified, it will be skipped and the input field will begin at that position instead.

fl (Field length). This defines the length of your input field. In can be a value between 1 and 240. INPUT@ will create a visible field of underline characters for this field. It will NOT allow you to overtype the field. Unless you set the "return on full field" option (described next), it will simply pause and refuse to except any more characters.

it (Item type). This controls what type of input will be allowed. You may use a literal or a string expression here. You have two options:

* "$" - Any alphanumeric characters
* "#" - Numeric characters only

"Numeric" characters are defined as:

0-9, decimal (.), plus (+), or minus (-)

By appending an asterisk to the field type specifier, you set the "return on full field" mode. That means when the last character in the field in entered, the statement proceeds. Otherwise, it will wait for an ENTER to be pressed to proceed. For example:

"$*" - Alphanumeric field, return when full.

If ENTER alone is pressed, you will be returned a null string. Also, there will not be a carriage return at the end of a string simply because of ENTER being used to terminate the input.

If the only thing that you press at the prompt is one of the function keys, it will abort at once and return you the value of that function key in the input string. Otherwise, if they are not the first item on the line, then they will simply be part of the received input string.

var$ (Return variable). This is a string variable that you specify for INPUT@ to return the input field to you in. It MUST be a string. Even if you input numeric only, it will still come to you as a string and you must get the VAL of it (see VAL in your Level II manual). This variable must be set off from the list by a semi colon. A comma will not work.

Examples

```
INPUT@512,"Type in your name: ",20,"$";NA$
```

This will print the defined string at screen position 512, and then print a 20 character field of underlines after it and accept any alphanumeric data into it. It will wait until ENTER is pressed to exit and will return the input data in "NA$".

```
INPUT@(10,30),40,"$*";SI$
```

This will not print a prompt string because one was not defined. With INPUT@, it is not necessary to leave in the extra comma. Simply ignore the prompt field if you don't wish to use it. It will print a 40 character field at screen position row 10, column 30 and terminate when the 40th character is typed.

The return on full field is useful when you are prompting for a single key entry and you wish to preclude the continual pressing of the ENTER key. You simply define a field length of one and to return when field full and as soon as they type a key, off you go.

While inputting data, you may use the following:

* Repeating keys.
* Backspace.
* Erase line (shift back arrow).

Note also that INPUT@ will not respond to JCL.

Label addressing (indirect branching within BASIC programs)

This function allows you to use indirect addressing within your BASIC programs. To accomplish this, we replaced the NAME function of BASIC with our own.

==========

NAME label
GOTO label
GOSUB label

> NAME assigns the specified label to the line on which the NAME statement appears. After that, you reference the label EXACTLY as you would a line number using GOTO and GOSUB statements.

==========

Labels may now be used in place of line numbers. This frees you from having to remember the exact line number that a particular subroutine was located at. Simply assign the subroutine a unique name and reference it by that.

The only restrictions are: (1) labels may NOT contain any reserved words when under OPTION S (see OPTION) and (2) labels may not exceed 240 characters in length.

We have also altered BASIC's RENUM function such that it will not regard labels when renumbering a program.

Labels may contain the letters A-Z, the numbers 0-9, @, or . (period). Labels may only begin with the letters A-Z or the "@" character. You should not use labels with the following BASIC statements:

```
ELSE
THEN
ERL
DELETE
RUN
```

That doesn't mean that you cannot use the statement "IF A=1 THEN GOTO TEST". The "GOTO" will set off the label. You should not, however, use "IF A=1 THEN TEST". Don't use labels directly with those statements.

Please note that a label must be the first statement on a line. For example:

```
10 NAME TEST:FOR A=1 TO 10
20 Other program here ...
100 GOTO TEST
```

The NAME statement is the first item on the line. If this is not the case, the name statement will be regarded as a comment and any attempt to reference it will result in an error.

Examples

```
10 CLEAR 1000 : DEFINT I
20 NAME START
Other program lines ...
1000 GOTO START
```

In this example, line 20 has been assigned the label "START". Later, at line 1000, the program issues the command to "GOTO START". This would send program control back to line 20.

```
10 NAME READINPUT
```

This is an example of an invalid label under OPTION S. This label contains the reserved words "READ" and "INPUT". BASIC will reject this as a label. Again, consult the enhancements to the OPTION command to learn about reserved words in labels.

Error messages (detailed error message display)

This feature of our BASIC Enhancements is not really a command in the usual sense, but rather is a manner in which the system functions that deserves to be documented.

When an error occurs under BASIC, the error message will be printed on the screen along with the offending statement. An arrow will identify the statement that contains the error.

For example:

```
10 FOR I=1 TO 10
20 PRINT "THIS IS A TEST",
30 X=C+2 : GOSUB : NEXT I
```

In this example, there is a "Undefined line number" error in line 30. In the middle of our loop, we issue a GOSUB with no line number or label.

The printout will appear something like this:

```
Undefined line number in 30
-->GOSUB : NEXT I
```

The arrow always points to the statement that contains the error, no matter how large the line. It does not point to the element within the statement that is incorrect. That is for you to determine.

OPTION (engage special options)

This command, present in standard BASIC, has been enhanced to allow Model III BASIC upward compatibility. It will remove the need for spaces around keywords.

==========

OPTION [param]

param is the optional indicator.

Your parameters are:

S	"Short form". Compatible with Model III BASIC.
L	"Long form". Standard Model 4 configuration. This is the default value.

==========

Standard Model 4 BASIC requires that you enter spaces around keywords. This is done so that you may use reserved words in long variables (and now labels, too). However, this also means that programs that have been written with the Model III may not convert easily to the Model 4.

Without the spaces, Model 4 BASIC doesn't know keywords. This allows keywords in variables, since without the spaces there is nothing special about the string of characters. We now allow you to alter this.

In the short form, reserved words themselves are once again significant with or without spaces. This is like the Model III. In the long form, of course, BASIC will react in the standard method.

These also cause the same effect on labels. Under OPTION S, you may not use a reserved word in a label. Under OPTION L, this is allowed.

This causes NO other changes. 40 character variable names are still valid. It only affects keyword recognition without spaces.

In order to transfer a program to Model 4 BASIC using this option, save the file in ASCII and transfer it (either directly or via CONV) to your TRSDOS 6.0. Enter BASIC in the standard manner and type:

OPTION S <ENTER>

Then load the file. The needed spaces will be inserted as the file is loading. Once the file is loaded, you may return to the standard form by typing:

OPTION L <ENTER>

You may switch back and forth at will.

Also, when under OPTION S, there is no need to include spaces as you are typing in the command line. For example, under OPTION L (the default), the line:

10 FORI=1TO10

will be seen as the variable "FORI" being equal to the value "1TO10". The same statement under OPTION S would expand to:

10 FOR I=1 TO 10

automatically. However, under OPTION S the label:

20 NAME READKEY

would become:

20 NAME READ KEY

and its value as a label would be lost because "READ" is a keyword. Under OPTION L, it would be left alone.

So you see, each method has advantages and disadvantages. However, without OPTION S, you could never load and run a program that did not already have spaces around the keywords. Therefore, if you use OPTION S for nothing but loading existing programs, it is STILL very useful.

RESOLVE (remove labels)

This utility allows you to remove label addressing from a program.

==========

SYSTEM"RESOLVE"

There are no parameters for this utility

==========

Label addressing is a great advantage when you are developing software and frees you from many of the contraints that having to use line numbers imposed. If you are going to continue to use this program under a BASIC enhanced with our options, then there will never be a need to remove the labels.

However, some of you will be using the labels for software development and then expecting this program to run under a BASIC not modified by our enhancements. This requires that there be some way to remove the labels.

RESOLVE will remove label addressing from a program and resolve all references to that label to the proper line number. It works completely in memory.

On the first pass, RESOLVE will turn all of the label references into line numbers. On the second pass, it will remove all of the NAME statements. If a NAME statement is the only item on a line, it will be replaced with a remark token.

Example

Assume that you had the following:

```
10 FOR I=1 TO 10
20 GOSUB TEST
30 NEXT I
40 END
50 NAME TEST
60 PRINT "Hello!"
70 RETURN
```

If you were to then execute:

```
SYSTEM"RESOLVE"
```

you would have:

```
10 FOR I=1 TO 10
20 GOSUB 50
30 NEXT I
40 END
50 '
60 PRINT "Hello!"
70 RETURN
```

Fast Lane Telecomputing with MTERM

Micro-System Software presents MTERM – the high speed smart terminal – for today and tomorrow's telecommunications. MTERM is the first truly high speed terminal program. Most versions can run at speeds up to 4800 baud without the insertion of null characters. And with the advent of 1200 baud modems and information services, this becomes an important consideration.

But MTERM is more than just a terminal program. It's a complete communications package. Supporting file transfer in both the traditional ASCII mode and the new "error-free" direct file mode.

Not only does MTERM surpass the competition in features offered, but it also offers ease of operation second to none. MTERM's powerful translation tables let you interface MTERM to any variety of communications service. Its unique MacroKey function allows you to have ten "user-defined" keys that transmit up to 64 characters at the touch of a key. MTERM even goes so far as to allow you to dial the phone and transmit the buffer at a specified time completely un-attended by the operator. MTERM is supplied with a user's manual that clearly explains program operation and answers many questions before they are asked.

MTERM will open up the world of telecommunications for you as never before. By using this program, you can do more in less time. The serious computer user cannot afford to be without a MTERM in their library.

And although MTERM geometrically outperforms the others, it's priced at only a fraction of their cost. MTERM is supported through the Micro-Systems technical support network, giving it support previously unavailable for a terminal program. Whether it is for the features, the performance, the documentation, the support, or the price, you just can't lose with MTERM.

CHECK THE FEATURES:

1. Spooled printer output and screen print function.
2. Easy to use translation tables with ready made default files for most applications.
3. Maximum auto dial support with user definable tables to allow dialing from a list of numbers at the touch of a key.
4. MacroKey function allows you to have up to ten "user-defined" keys with a maximum of 64 characters per key. Can be used for multiple "auto-logons" or any of hundreds of other applications.
5. Return to menu and execute MTERM command while still receiving data. Screen will be updated upon return. Never any lost data with MTERM!
6. Supports file transfer with either the traditional ASCII mode or new "error-free" direct file mode.
7. DOS commands from menu without exiting program.
8. Large capture buffer for uploading and downloading.
9. MTERM supports most major brands of modems, with full auto-dial capabilities where applicable.
10. MTERM fully supports some of these modem's advanced features that the competitors won't touch.
11. Simple and easy to use program with clear users' manual.

Does Your Dealer have the full line of Micro-Systems Software Programs?

Check these other fine products from Micro-Systems Software:

For Your TRS-80 Models I, III, and 4:
DOSPLUS – Advanced but user-friendly Disk Operating Systems • MTERM – Smart terminal program • BBS-80 – Ultra sophisticated Bulletin Board System

For Your TRS-80 Models II and 16:
MTERM – Smart terminal program

For Your IBM Personal Computer, Apple II, and Zenith Z-100:
MTERM – Smart terminal program

Micro-Systems Software Inc.
4301-18 Oak Circle
Boca Raton, FL 33431
Sales: (800) 327-8724
Technical: (305) 983-3390
Telex: 467384 MSS INC CI

Micro-Systems Software enthusiastically introduces the release of 6.0 PLUS, a series of advanced enhancements to TRSDOS version 6 for the TRS-80 Model 4. Finally there is a package that will enable the novice and expert alike to increase their programming speed and ability above the limits created by TRSDOS 6.0. This enhancement package is titled 6.0 PLUS.

6.0 PLUS is equipped with a series of BASIC enhancements that make any programmers life easier. Now possible with this package will be multi-array sorting, comprehensive cross referencing, and the ability to search and replace BASIC text. Also included will be label addressing, more descriptive error messages, and an expanded OPTION command. Have you thought about all the time necessary to convert programs from Model III BASIC to Model 4 BASIC? Well, 6.0 PLUS will allow you to load Model III disk BASIC programs into the native Model 4 disk BASIC and do all keyword space insertion, saving you hours of valuable time and effort. Additions to the OPTION command provide for this compatibility with Model III Disk BASIC. Screen formatting for attractive applications software display are permitted by Micro-Systems Software's exclusive INPUT@ keyword. Think about what all these features will mean for you.

Have you ever had the misfortune of losing all the data on your disk for some unknown reason or just because your power went off for a split second? You probably wished there was some way to repair that blown byte or sector. 6.0 PLUS will aid you in repairing these annoying errors. Included in the enhancement package is a full range of disk management programs including a disk editor, a file editor, and a directory verification/repair utility. Direct access is permitted to any track of the disk, which allows you to correct any byte that may be in error. Even if the unfortunate victim on your disk happened to be your directory, 6.0 PLUS will rebuild it. 6.0 PLUS can mean the difference between losing a track's worth of information or an entire disk.

We have all at one time or another made the mistake of REMOVing the wrong file. Once this is accomplished you know you can look forward to the wonderful task of rewriting the whole non-existant thing. But wait! No problem here either, just use the RESTORE command. Error-proofed RESTORE has the ability to rebuild this lost file as long as the disk space has not been reallocated. You may never lose another file again in this manner.

6.0 PLUS is a "must have" and a valuable "toolbox" for the programmer. You will never understand how you ever could have lived without the enhancements after using them. This package will be one of the most valuable pieces of software you have ever purchased and also one of the most inexpensive. 6.0 PLUS retails for a very low $49.95. This package allows the TRSDOS user to utilize some of the powerful utilities of DOSPLUS IV without having to purchase an entire operating system. Micro-Systems Software created 6.0 PLUS for the TRSDOS user that wants more than just an ordinary operating system. 6.0 PLUS gives you the power you strive for.

INTRODUCTION

This document accompanies the update to 6.0 PLUS that permits this software to work with TRSDOS 6.2 and BASIC 1.1. This is the only alteration to the documentation. Any information contained in this supplement will supercede the corresponding information in the old manual. Apart from that, nothing has changed.

The actual OPERATION of the software has altered little or none at all. For the most part, this is because the version of TRSDOS did not matter. It was the version of BASIC that spelled the problem. The new BASIC 1.1 was a fairly extensive re-working of the prior version, and a great deal of the code was moved. Therefore, although the DOS utilities were compatible, the update required that we dis-assemble BASIC again and re-write the BASIC enhancments 100%. This spelled many hours of work.

And you have the results of that work in your hands now. A completed 6.2 PLUS package. DOS and BASIC enhancements for the TRSDOS 6 operating system. We certainly hope that you enjoy these programs. Please feel free to write us with your comments or suggestions. Now, let's cover the specifics of the update. What HAS changed?

THE MASTER DISKETTE

TRSDOS 6.2 takes up more space on a diskette than did the prior version. Because of this, there was no space for the enhancements to be delivered on a system diskette. Hence, the 6.2 PLUS package is sent out on a formatted data diskette. Use the TRSDOS COPY command to move what modules you wish to a system diskette. Bear in mind that once you make the decision between the two BASIC enhancment files (BE1 and BE2), you do not need both of them. Also, as it states in the manual, once you have INSTALLED the BASIC enhancements, they are part of the file BASIC/CMD. You no longer need the BE1 or BE2 files on the diskette at all.

MODIFYING BASIC/CMD

The installation for the BASIC programming enhancements has changed just slightly between versions. Refer to page 31 of the manual, to the top part of the page. The commands that are there showing you how to use the DUMP command to create the modified BASIC/CMD are now slightly incorrect. Be CERTAIN that you still load BASIC/CMD first, as shown at the bottom of page 30.

In the example for the BE1 file, the value for END should be X'8791' and TRA should be X'8543'. So the command becomes:

```
DUMP BASIC/CMD.BASIC (START=X'3000',END=X'8791',TRA=X'8543')
```

In the example for the BE2 file, the value for END should be X'8673' and TRA should be X'8425'. So the command becomes:

```
DUMP BASIC/CMD.BASIC (START=X'3000',END=X'8673',TRA=X'8425')
```

Other than these two instances, there are NO functional, installation, or operational differences between 6.0 PLUS and 6.2 PLUS. The newer programs are updates for the later version of BASIC.